The invention takes advantage of patterns in the state space, thereby making possible the verification of a wider class of machines, as well as improved the overall speed of the verification. In the preferred embodiment, polymorphism provided a powerful mechanism for allowing the comparison of generic machines, which was realized by treating each instance of polymorphism as a single pattern in the state space.

4. Other Embodiments

It should be understood that the invention is not limited to the embodiment described above but that it covers other embodiments. For example, data types, which are determined through static analysis in the above embodiment, may be determined through other analysis methods or manual operation. The information specified by generic queries may be entered through manual operation or other analysis methods. The program logic comparator, though implemented on a computer in the above embodiment, may be implemented, in whole or part, by an electronic circuit.

5. Effect of the Invention

The present invention provides a logic program comparison method which makes it possible to do verification by comparing parameterized logic programs and which increases the efficiency of the verification.

What is claimed is:

1. A computer-based logic program verification method comprising:

receiving two logic programs, each having procedures with program variables, each of said program variables being of a corresponding data type;

converting said logic programs to a first and second finite state machine description respectively, each having internal states, input values and output values, comprising,

determining the data types of variables of said logic programs,

converting said programs to a completed form having expressions in disjunctive form,

expanding procedure calls within said programs into corresponding procedure bodies,

translating said programs into corresponding transition functions composed of transition function expressions;

determining from said transition functions whether there exists an equivalence between internal states, between input values, and between output values of said first and second finite state machine descriptions, and determining whether said finite state machine descriptions produce respective output values deemed equal for all respective inputs deemed equal and for all respective states deemed equal; and

outputting the result of said comparing step.

2. A computer-based logic program verification method according to claim **1**, wherein said receiving step receives generic queries as part of said programs and wherein said converting step performs said converting based on said generic queries.

3. A computer-based logic program verification method according to claim **1**, wherein said converting step determines said data types by applying static analysis to said programs.

4. A computer-based logic program verification method according to claim **1**, wherein said comparing step uses a constraint solving technique comprising the steps of:

associating a unique constraint variable with each program variable denoting an input, output, or state variable;

grouping each constraint variable of a particular data type from said first logic program with every other constraint variable of a corresponding data type from said first logic program to form pairs;

grouping each constraint variable of a particular data type from said second logic program with every other constraint variable of a corresponding data type from said second logic program to form pairs;

generating a conjunction of inequalities between each pair of said constraint variables;

generating, for all possible reorderings of the expressions in disjunctive form equalities between constraint variables occurring in respective positions;

taking the conjunction of inequalities and said generated equalities as constraints to a new expression;

solving the new expression using the constraints.

5. A computer-based logic program verification method according to claim **1**, wherein said logic programs are described by definite clauses that are a subset of predicate logic, and wherein a definite clause is restricted to one of a query, a rule and a fact.

6. A computer-based logic program verification method according to claim **1**, wherein recursive procedures having recursive procedure bodies are mapped into FSMs by mapping the recursive procedure bodies into corresponding iterative expressions, thus removing corresponding recursive calls.

7. A logic program verification method according to claim **1**, wherein said converting step further comprises:

permuting the transition function expressions to group together variables with common values, and

replacing each variable by a unique code;

permuting the transition function expressions to group together variables with common values;

converting transition function expressions where one of said internal states takes on multiple values into multiple expressions wherein said one of said internal states takes on a different corresponding value in each of said multiple expressions;

encoding each transition function expression having a unique internal state with a unique code.

* * * * *